getting started with git

todd mcleod

version control software

training

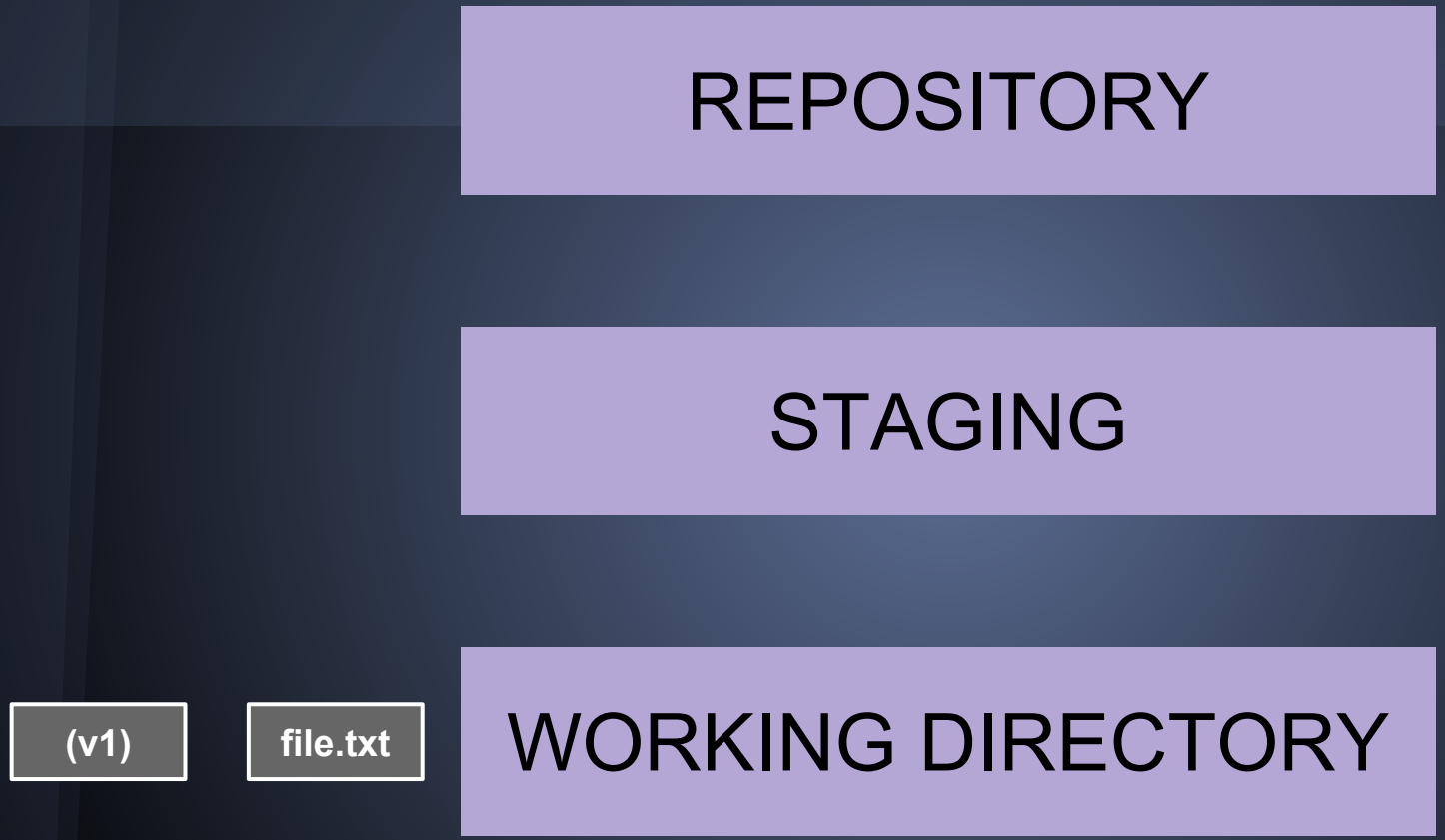
source code management

git vs GitHub

REPOSITORY

STAGING

WORKING DIRECTORY


REPOSITORY
STAGING
WORKING DIRECTORY
(v1)
file.txt

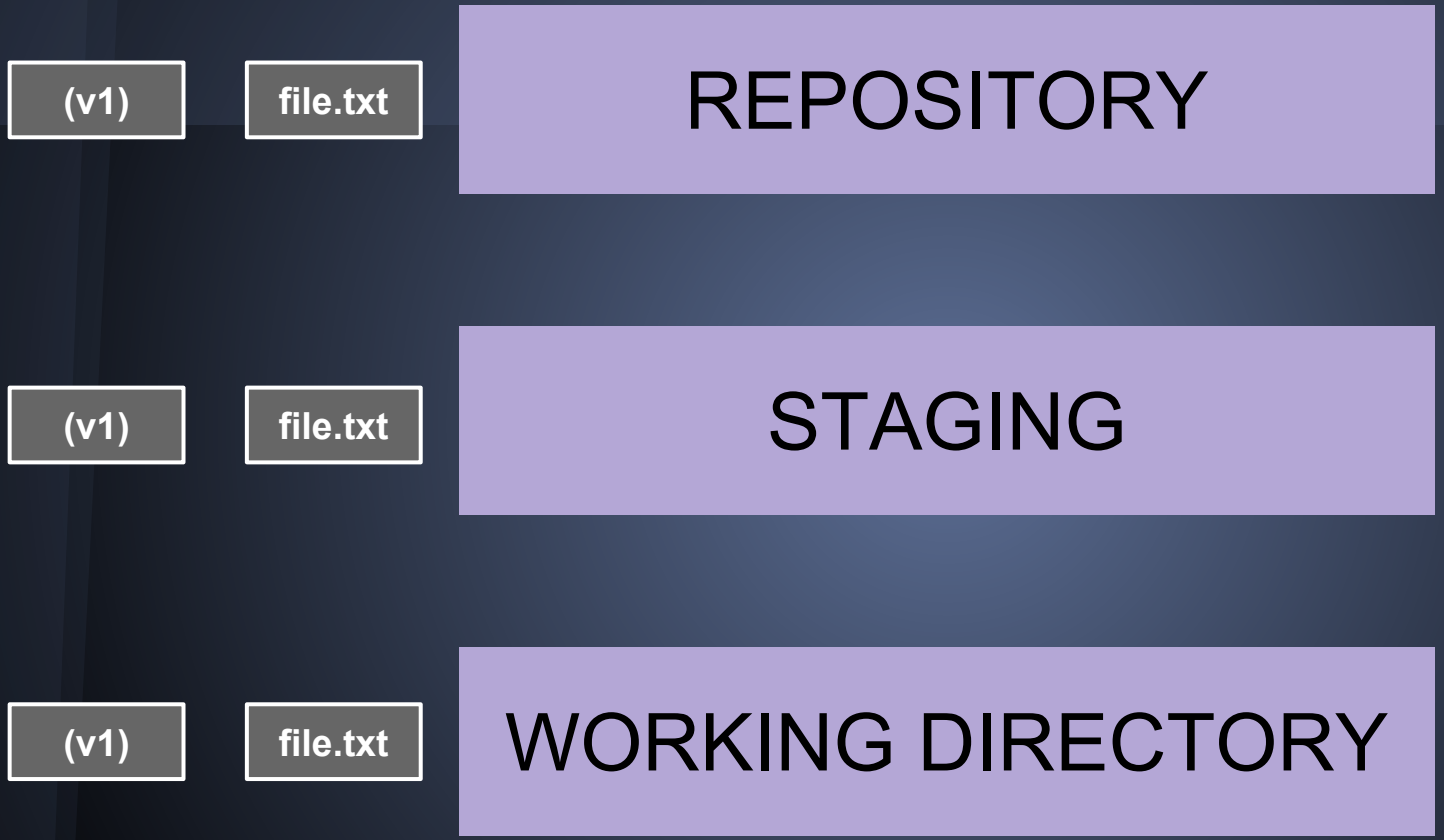
REPOSITORY

(v1) file.txt

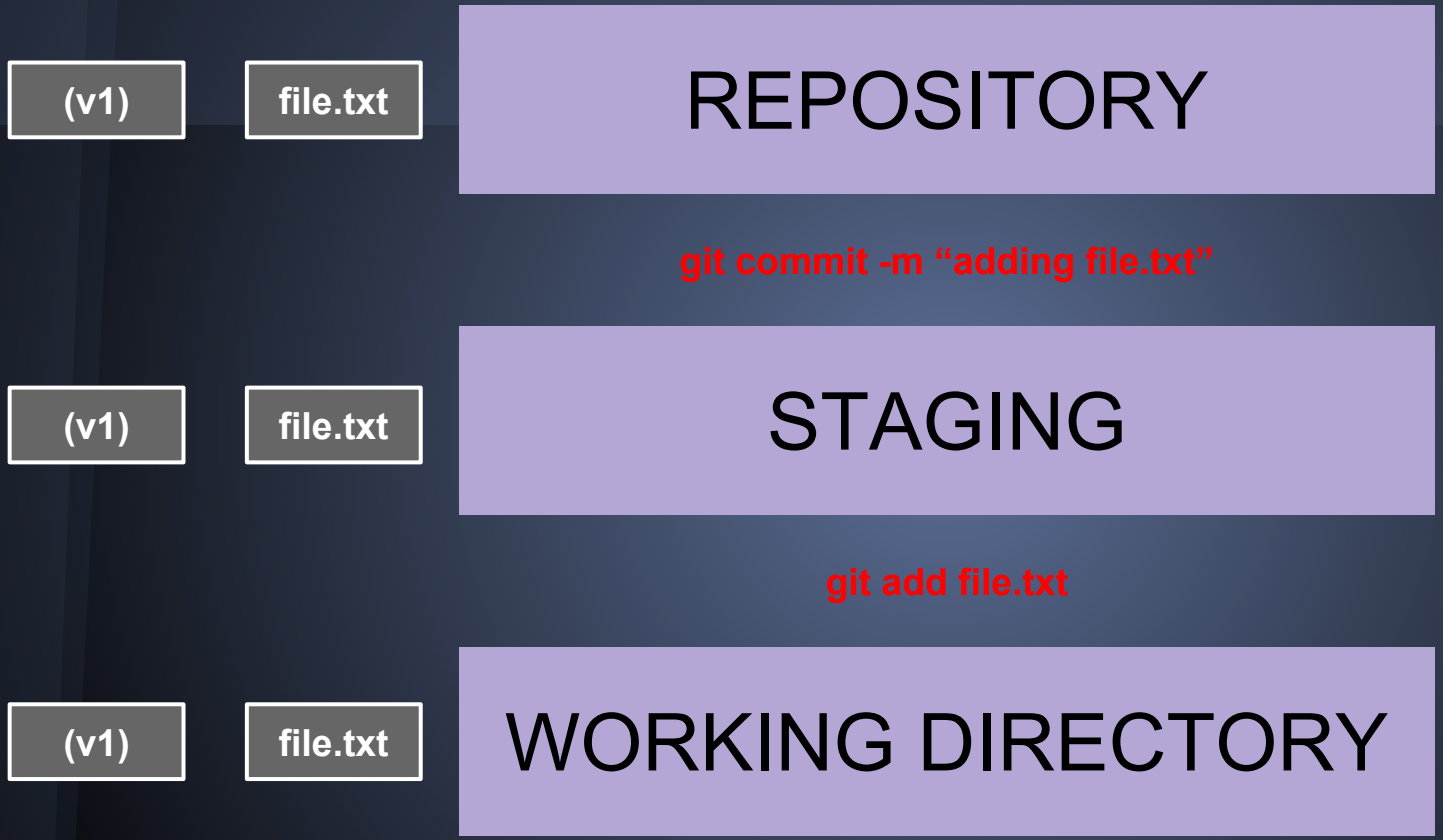
STAGING

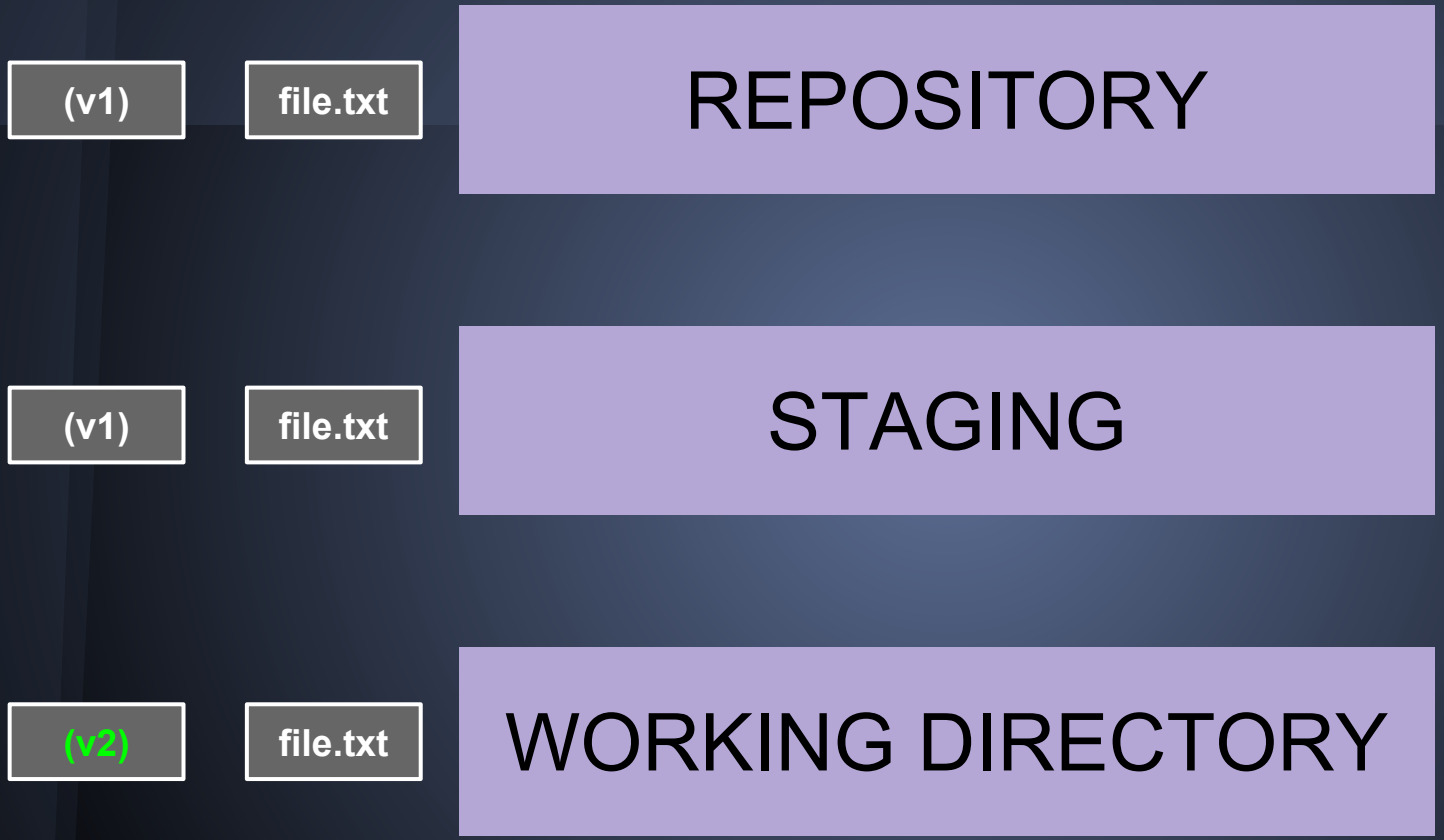
(v1) file.txt

WORKING DIRECTORY


(v1)
file.txt
REPOSITORY
(v1)
file.txt
STAGING
(v1)
file.txt
WORKING DIRECTORY


(v1)
file.txt
REPOSITORY
(v1)
file.txt
STAGING
git add file.txt
(v1)
file.txt
WORKING DIRECTORY


(v1)
file.txt
REPOSITORY
git commit -m “adding file.txt”
(v1)
file.txt
STAGING
git add file.txt
(v1)
file.txt
WORKING DIRECTORY

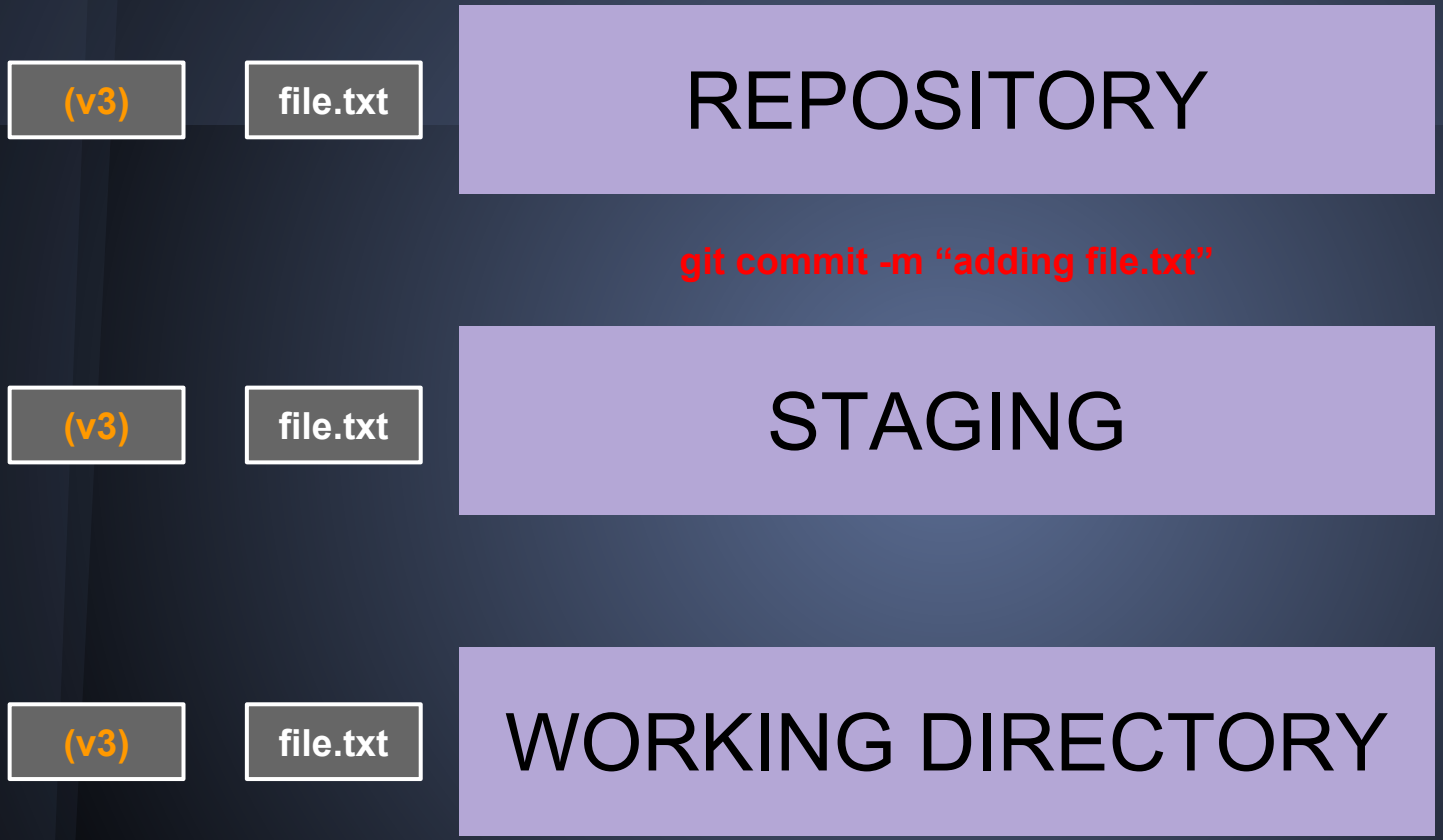

(v1)
file.txt
REPOSITORY
(v1)
file.txt
STAGING
(v2)
file.txt
WORKING DIRECTORY

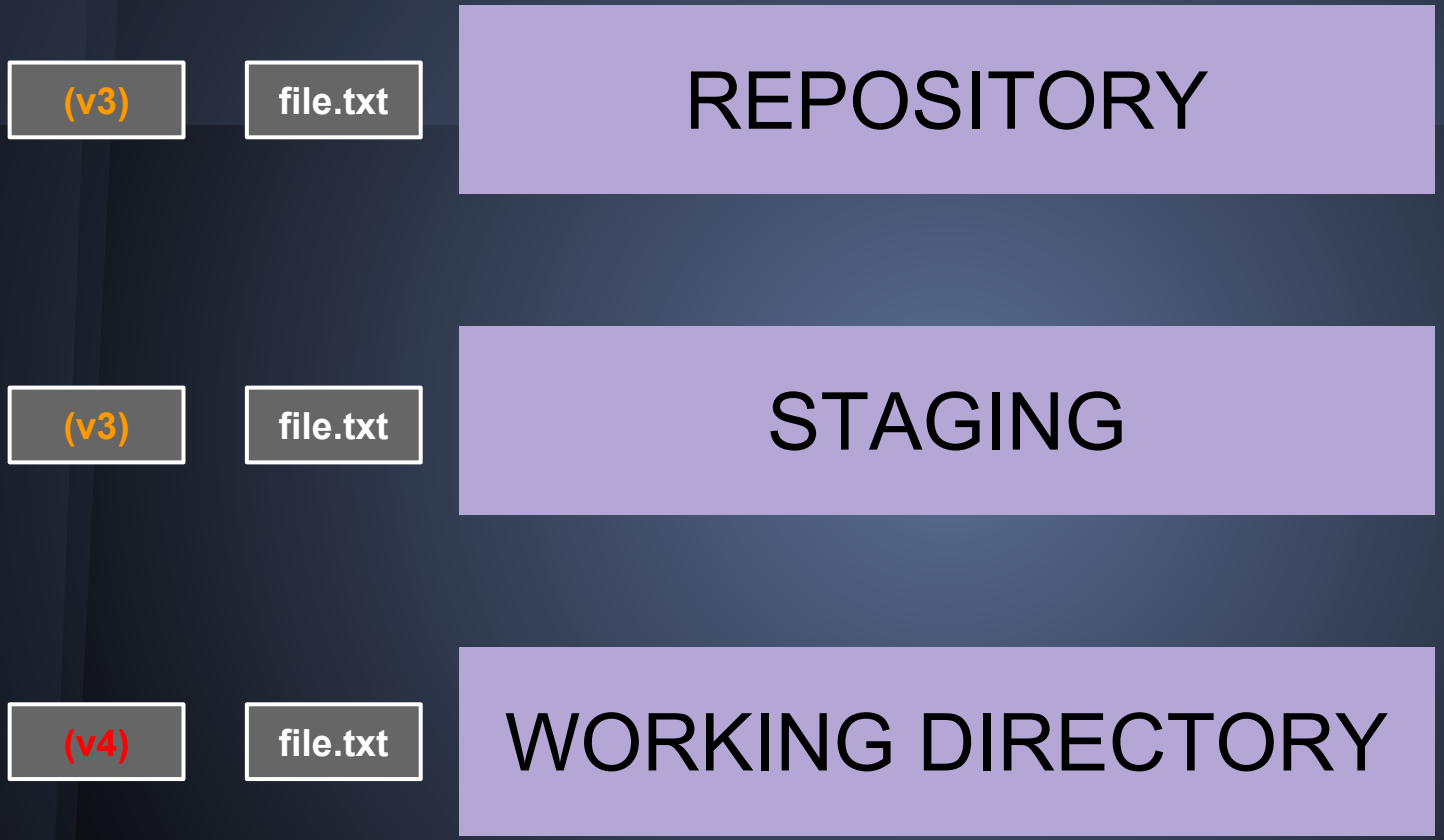

(v1)
file.txt
REPOSITORY
(v2)
file.txt
STAGING
git add file.txt
(v2)
file.txt
WORKING DIRECTORY

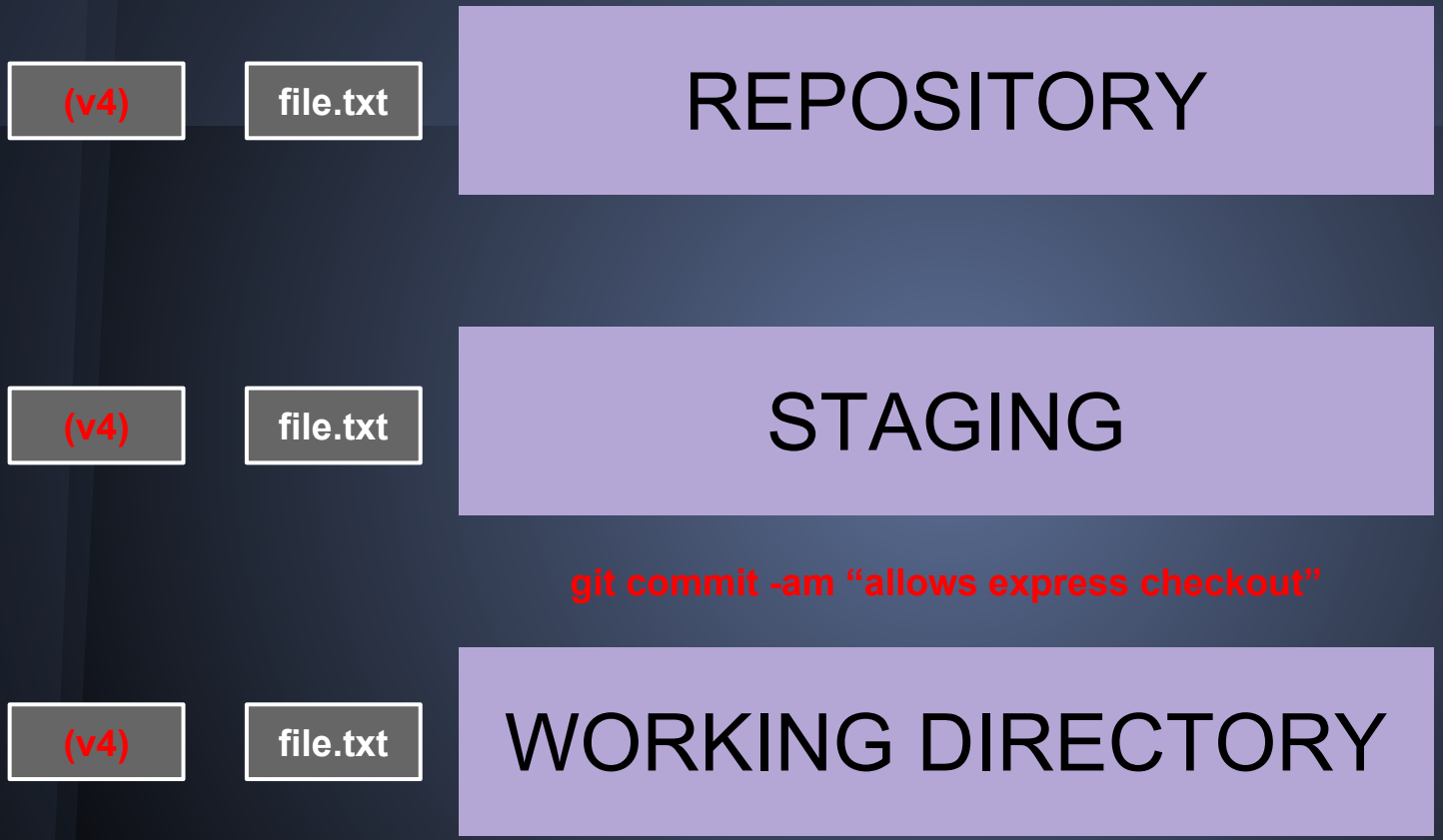

(v2)
file.txt
REPOSITORY
git commit -m “Redirect user to profile page after login”
(v2)
file.txt
STAGING
(v2)
file.txt
WORKING DIRECTORY


(v2)
file.txt
REPOSITORY
(v2)
file.txt
STAGING
(v3)
file.txt
WORKING DIRECTORY


(v2)
file.txt
REPOSITORY
(v3)
file.txt
STAGING
git add file.txt
(v3)
file.txt
WORKING DIRECTORY


(v3)
file.txt
REPOSITORY
git commit -m “adding file.txt”
(v3)
file.txt
STAGING
(v3)
file.txt
WORKING DIRECTORY


(v3)
file.txt
REPOSITORY
(v3)
file.txt
STAGING
(v4)
file.txt
WORKING DIRECTORY

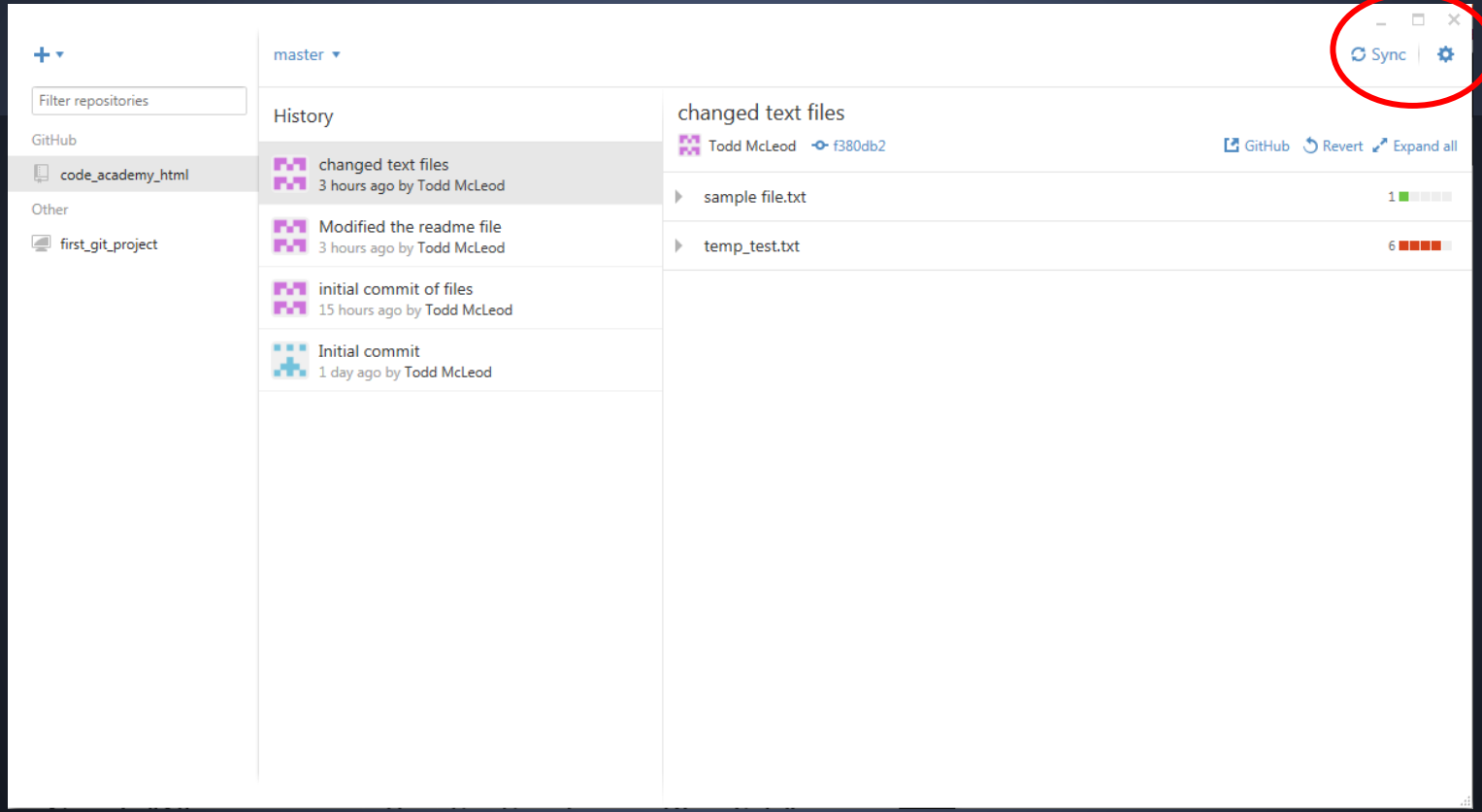

(v4)
file.txt
REPOSITORY
(v4)
file.txt
STAGING
git commit -am “allows express checkout”
(v4)
file.txt
WORKING DIRECTORY

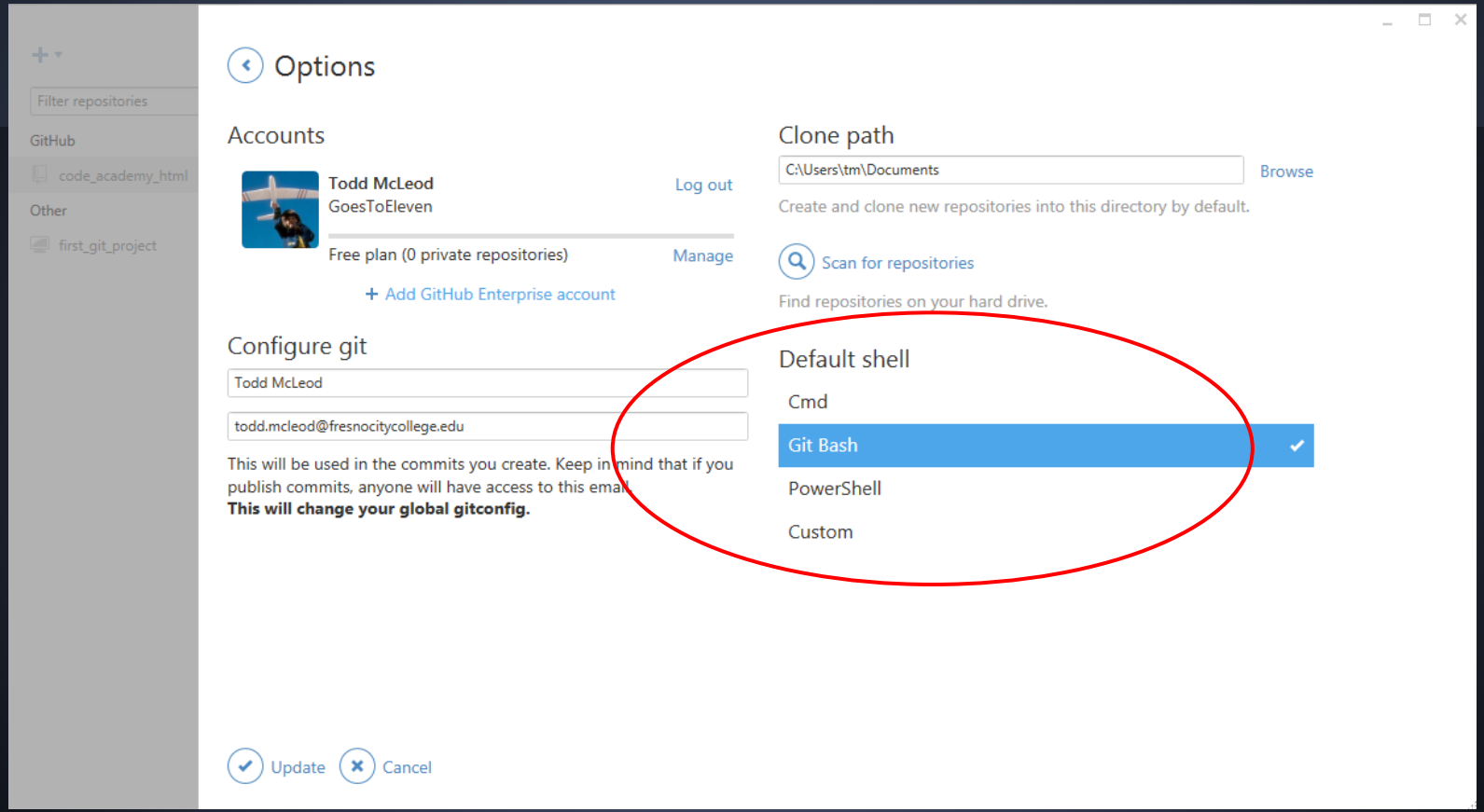
workflow

1. *make changes*
2. *add changes*
3. *commit changes*

GitHub

download & install


master
Sync
Filter repositories
GitHub
code_academy_html
Other
first_git_project
History
changed text files
3 hours ago by Todd McLeod
Modified the readme file
3 hours ago by Todd McLeod
initial commit of files
15 hours ago by Todd McLeod
Initial commit
1 day ago by Todd McLeod
changed text files
Todd McLeod
f380db2
GitHub
Revert
Expand all
sample file.txt
1
temp_test.txt
6


Filter repositories
GitHub
code_academy_html
Other
first_git_project
Options
Accounts
Todd McLeod
GoesToEleven
Log out
Free plan (0 private repositories)
Manage
+ Add GitHub Enterprise account
Configure git
Todd McLeod
todd.mcleod@fresnocitycollege.edu
This will be used in the commits you create. Keep in mind that if you publish commits, anyone will have access to this ema
This will change your global gitconfig.
Clone path
C:\Users\tm\Documents
Browse
Create and clone new repositories into this directory by default.
Scan for repositories
Find repositories on your hard drive.
Default shell
Cmd
Git Bash
PowerShell
Custom
Update
Cancel

git init

initialize directory

workflow

1. *make changes*
2. *add changes*
3. *commit changes*

Now we can do this!

git status

reports difference between our
working directory, staging index, & repository

git diff

compares repository with working directory

(more accurately: with where HEAD is pointing or with parameter)

git diff

git diff <file_name>

git diff --staged

git diff <commit_hash_code>

diff between working directory and the commit referenced by the hash code

git diff <commit_hash_code> <file_name>

git diff HEAD ^^

viewing a file

OS

file management

viewing a file

cat <file_name>

unix / linux command
‘concatenate’

viewing a file

git show <tree-ish>

git ls-tree HEAD

git log

shows commit log

git log

git log --oneline

git log --oneline -3

git log -n 2

git log --since=2013-03-12

git log --until=2013-12-31

git log --since=2013-03-12 --until=2013-12-31

git log --author="Todd"

git log --grep="Init"

git help log

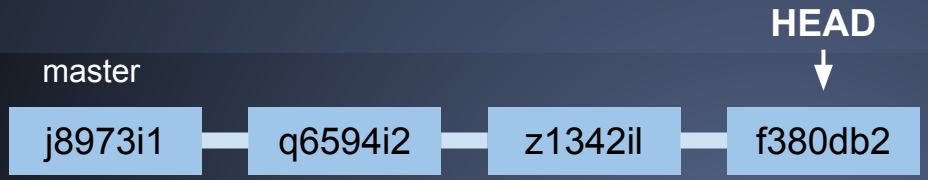
git help <command>

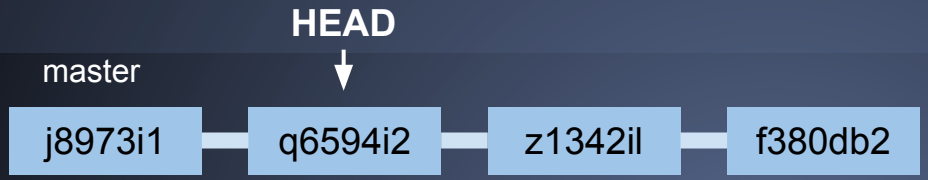
hash values

identify each commit

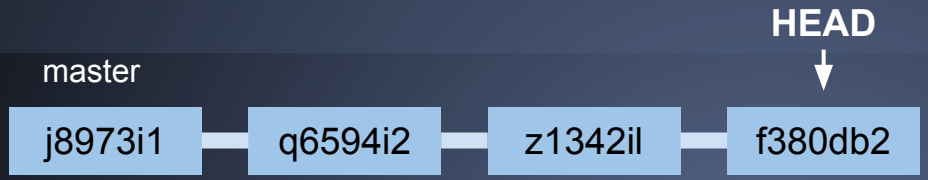
HEAD

‘playhead’

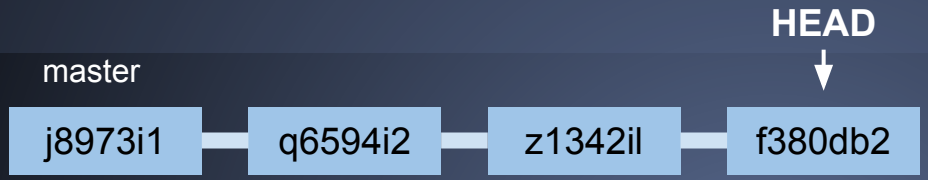

HEAD
master
j8973i1
q6594i2
z1342il
f380db2



git reset HEAD~2



git reset f380db2



git reset ORIG_HEAD

HEAD

```
cd .git
cat HEAD
cd refs
cd heads
cat master
git log
```

git checkout

revert to version in repository

git checkout

git show <commit_hash_code>

google: git show file in repository

git ls-tree -r HEAD

git show <file_hash_code>

git checkout <file_name>

git rm <file_name>

deletes a file

git mv <old> <new>

renames a file

git mv

git mv <old_file_name> <new_file_name>

git status

git log

Renaming through OS

rename file

git status

git add <new_file_name>

git rm <old_file_name>

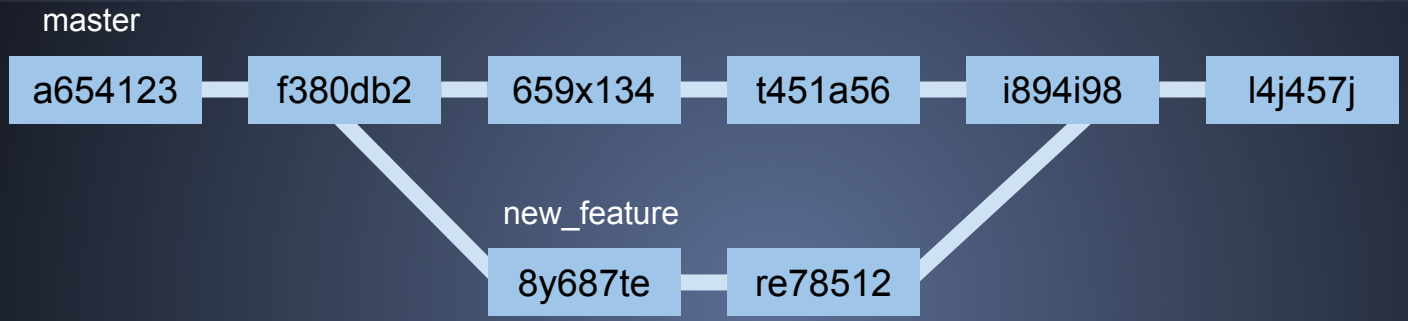
git status

git commit -m “renamed <old_file_name> to <new_file_name>”

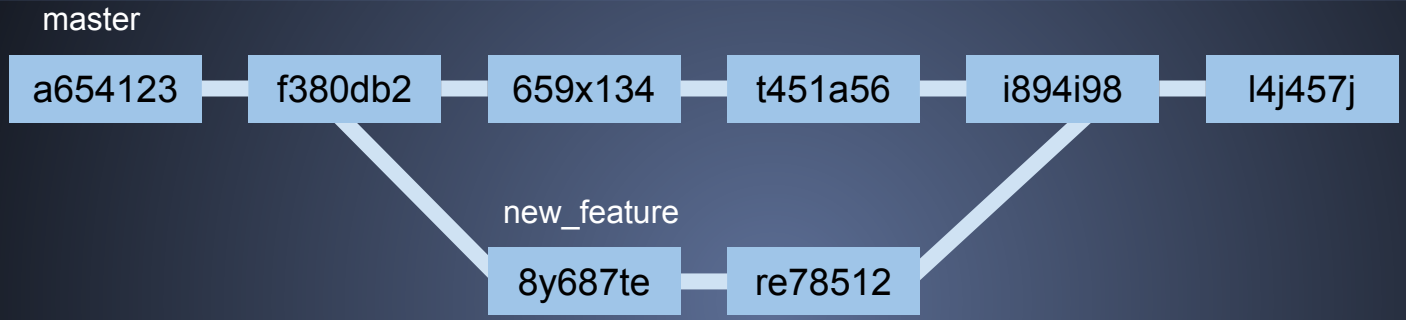
branching

try something; collaborate

branching

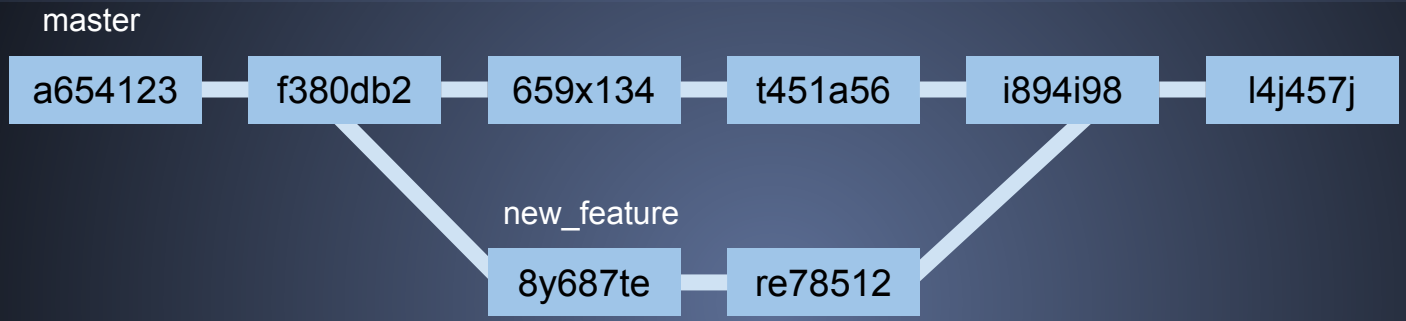

master
a654123
f380db2
659x134
t451a56
i894i98
l4j457j
new_feature
8y687te
re78512

branching - try something



- you have an idea
- instead of trying it, making commits, then undoing it
- create a new branch, try your ideas there
  - doesn't work, throw away the branch
  - does work, merge your branch back into the master branch

branching - collaborate



- one master branch
- create a branch to work on the project
  - doesn't work, throw away the branch
  - does work, merge your branch back into the master branch

git branch

shows branches

git branch <some_name>

creates a branch

git checkout <branch>

switches to branch

git log --graph --oneline --decorate --all

display log of branches

git branch --merged

shows identical branches

git merge <branch>

merges <branch> into current branch

(current branch = the branch that is currently checked out)

merge conflicts

open up the file, fix conflicts, commit

git log --graph --oneline --decorate --all

display log of branches

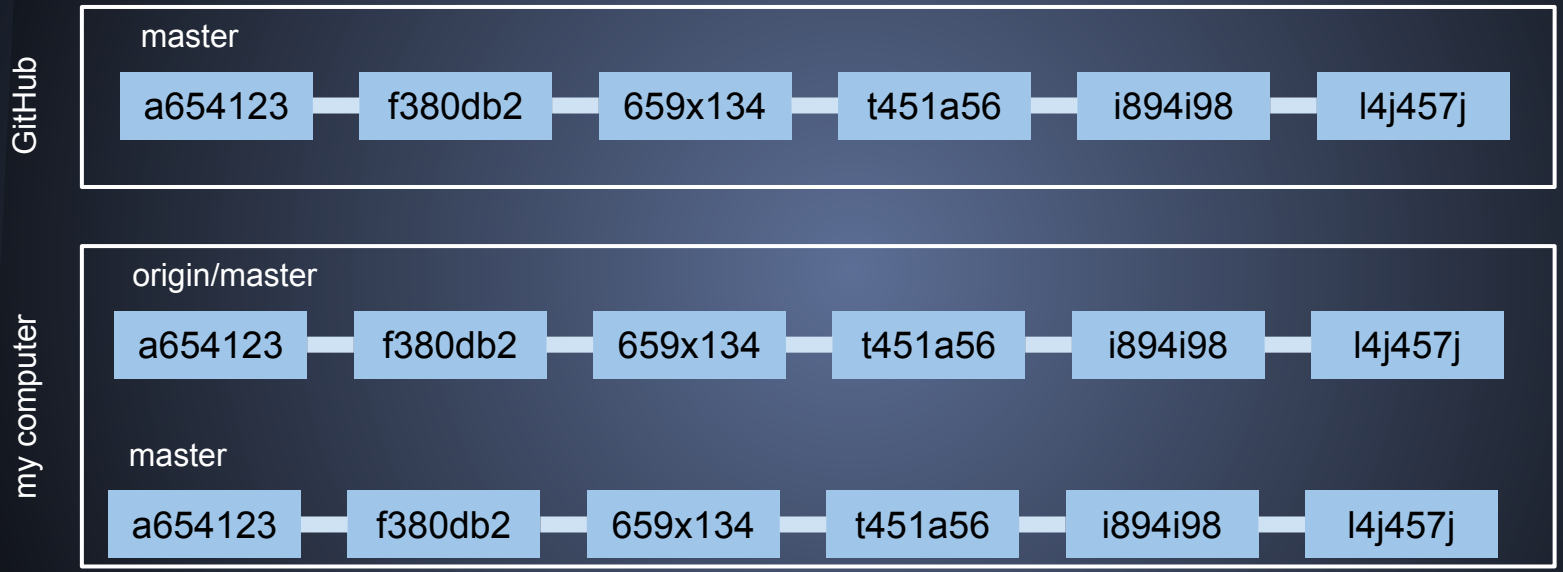
git branch -d <branch>

deletes a branch

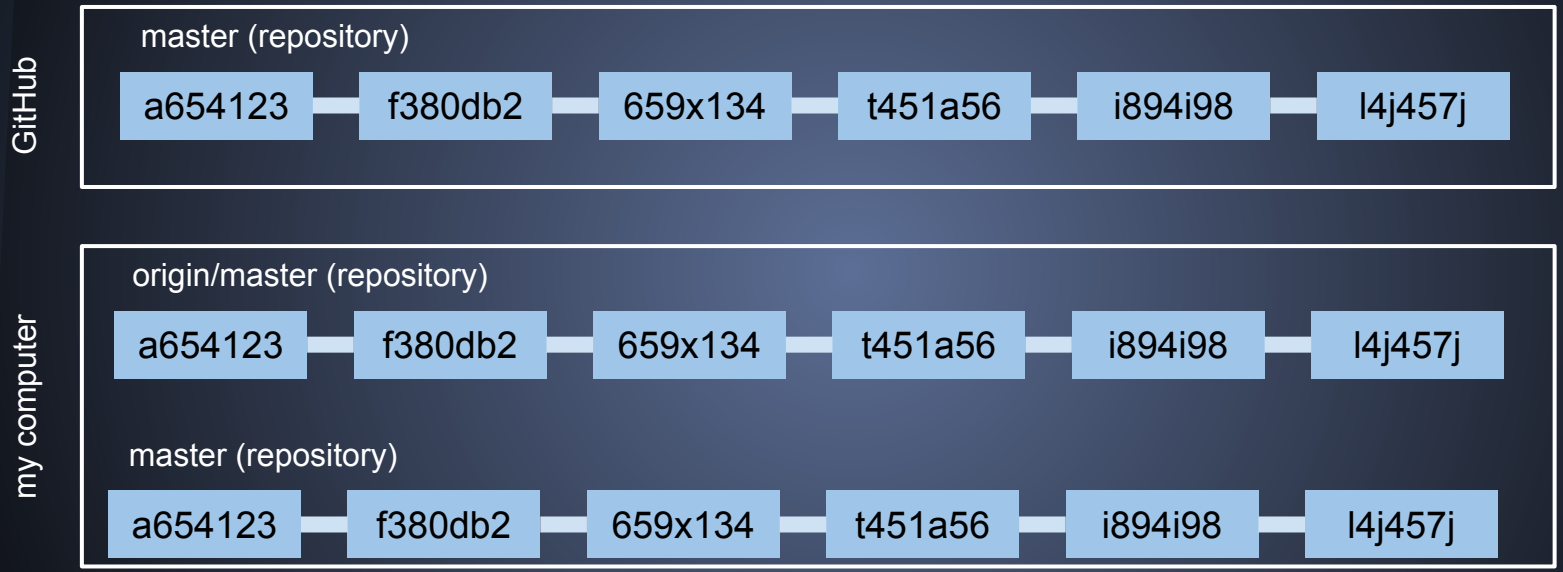
GitHub

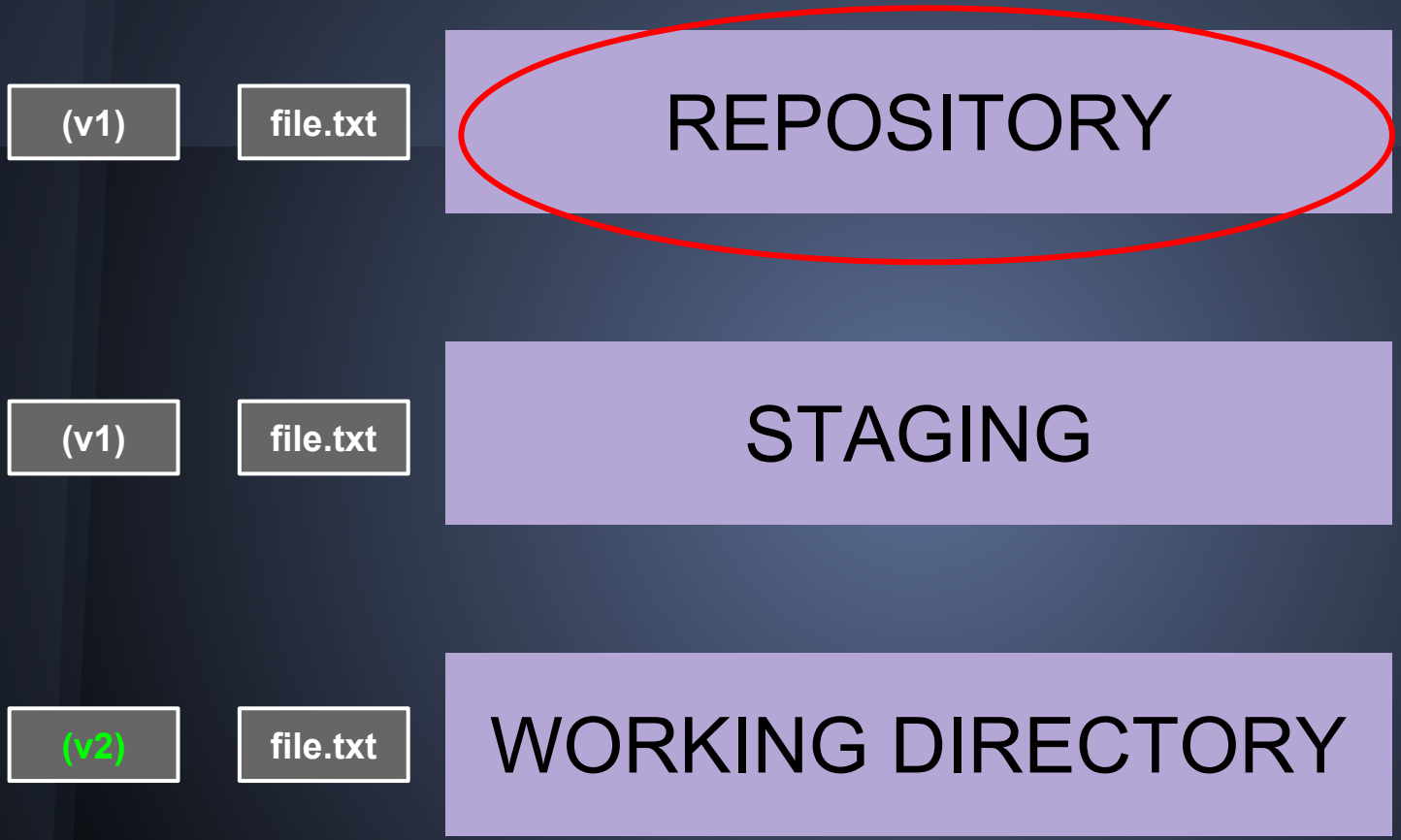
distributed version control

GitHub

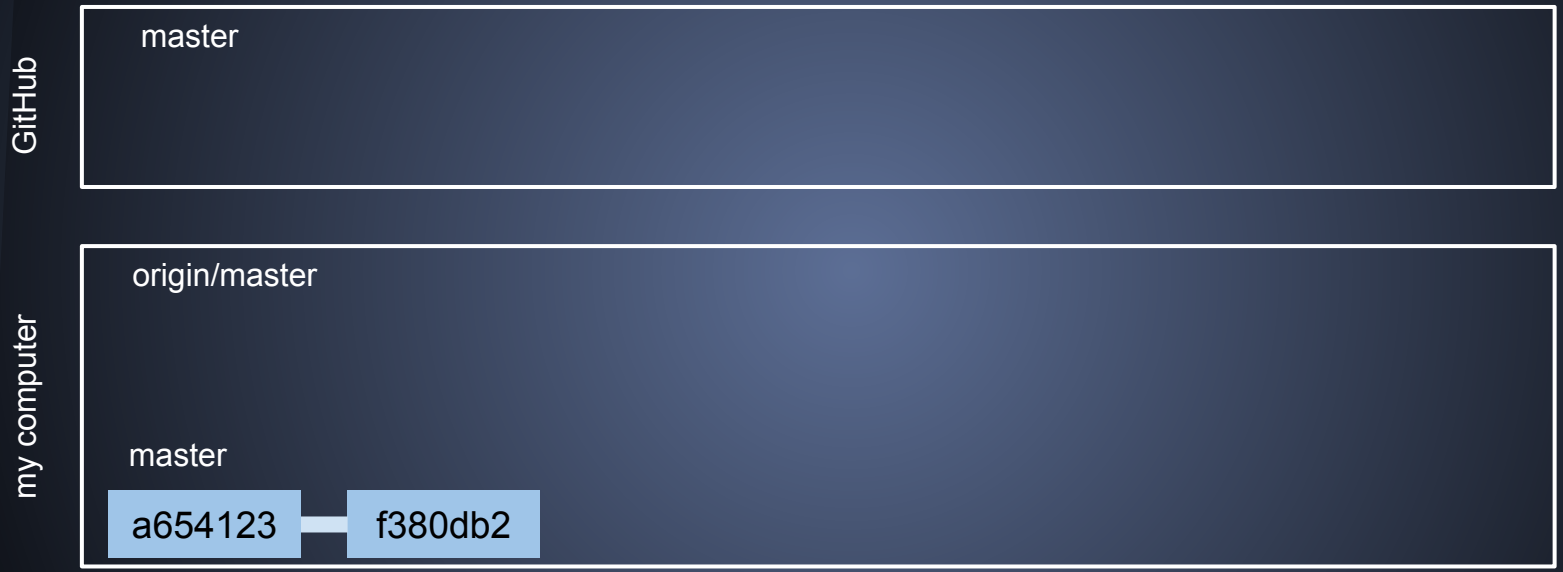

GitHub
master
a654123
f380db2
659x134
t451a56
i894i98
l4j457j
my computer
origin/master
a654123
f380db2
659x134
t451a56
i894i98
l4j457j
master
a654123
f380db2
659x134
t451a56
i894i98
l4j457j

GitHub

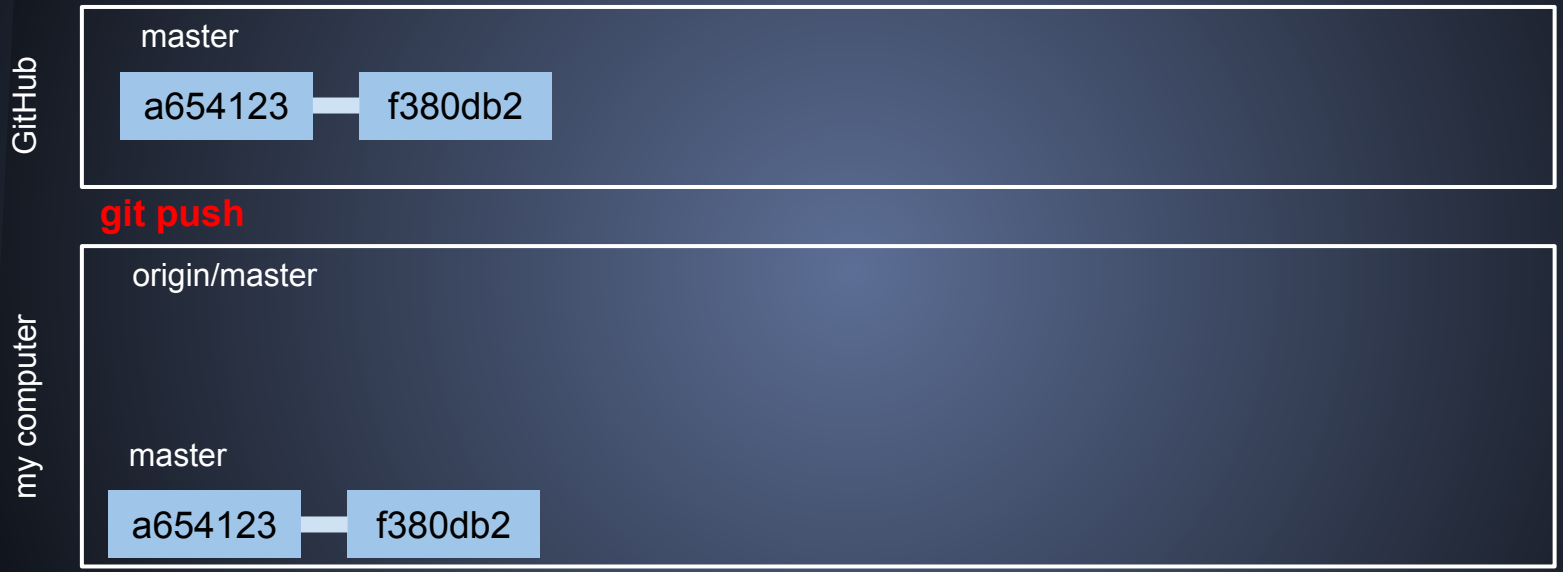

GitHub
master (repository)
a654123
f380db2
659x134
t451a56
i894i98
l4j457j
my computer
origin/master (repository)
a654123
f380db2
659x134
t451a56
i894i98
l4j457j
master (repository)
a654123
f380db2
659x134
t451a56
i894i98
l4j457j


(v1)
file.txt
REPOSITORY
(v1)
file.txt
STAGING
(v2)
file.txt
WORKING DIRECTORY

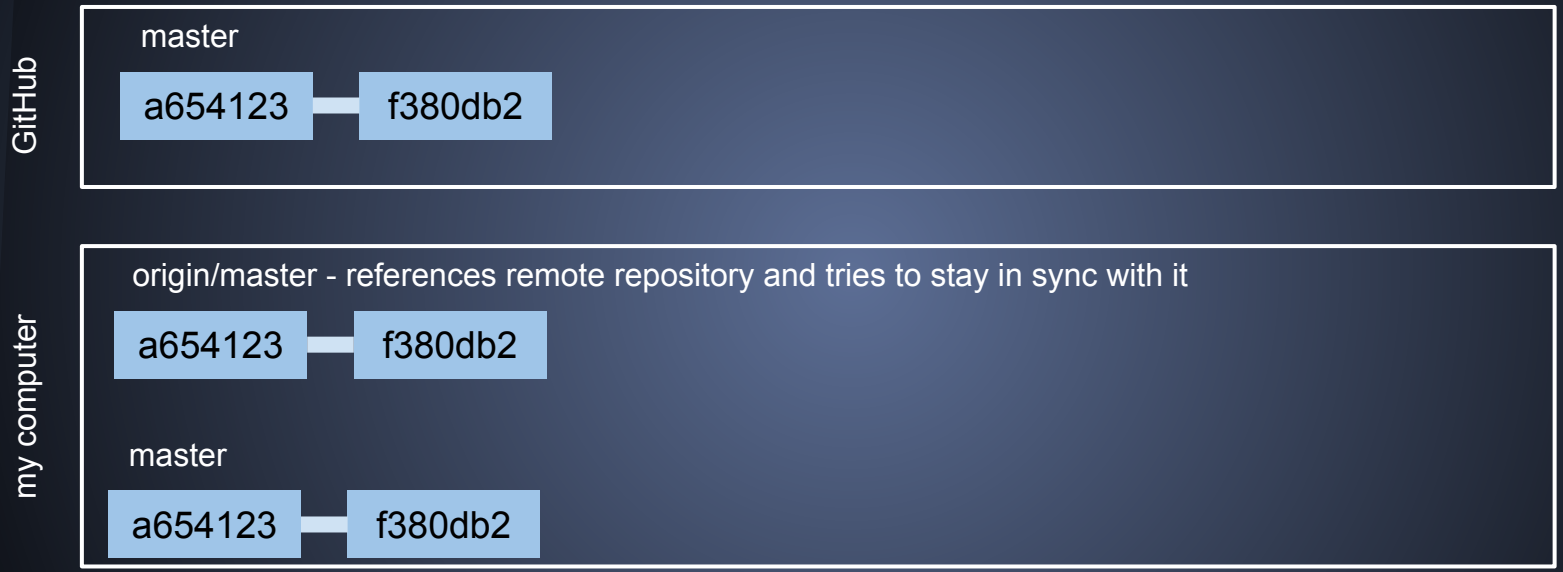
GitHub


GitHub
master
my computer
origin/master
master
a654123
f380db2

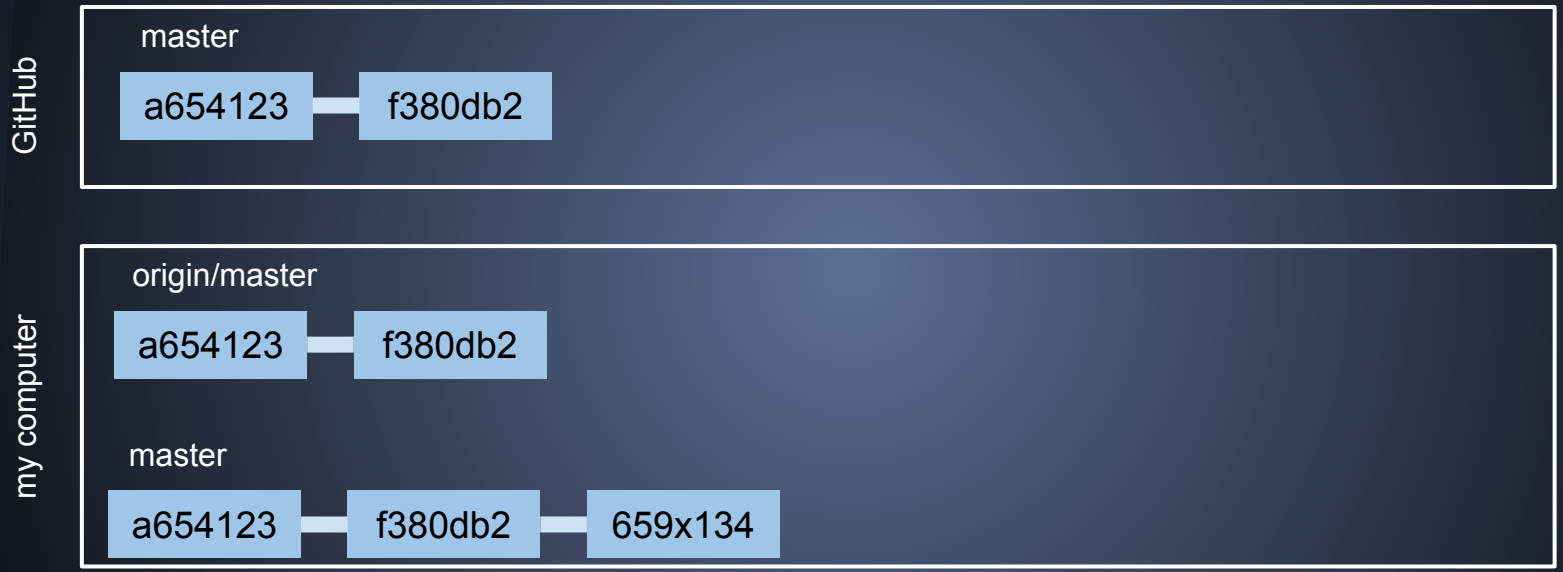
GitHub


GitHub
master
a654123
f380db2
git push
my computer
origin/master
master
a654123
f380db2

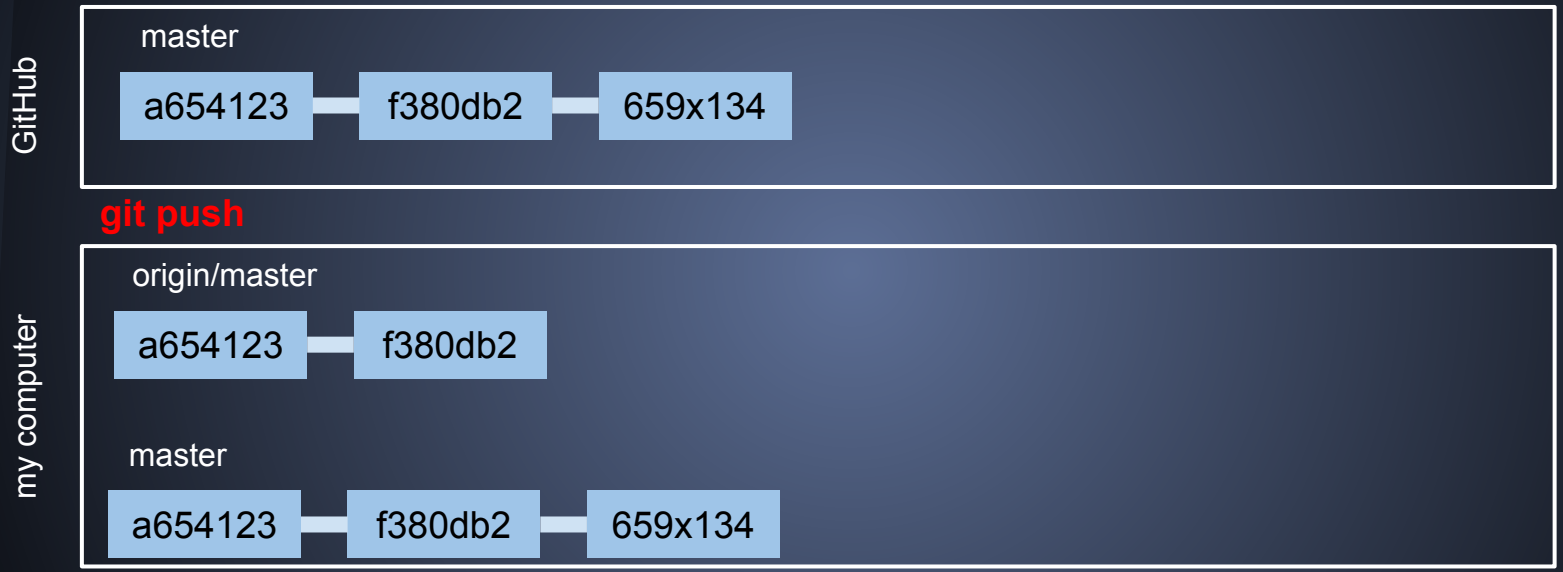
GitHub


GitHub
master
a654123
f380db2
my computer
origin/master - references remote repository and tries to stay in sync with it
a654123
f380db2
master
a654123
f380db2

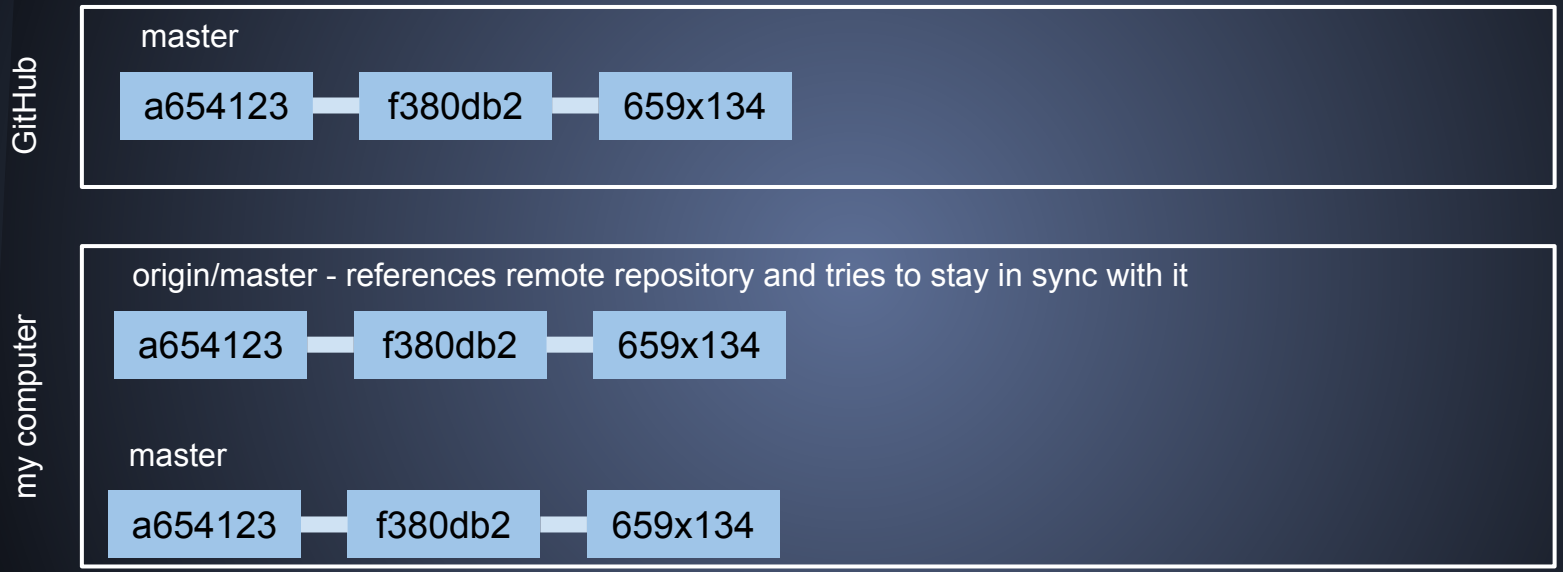
GitHub


GitHub
master
a654123
f380db2
my computer
origin/master
a654123
f380db2
master
a654123
f380db2
659x134

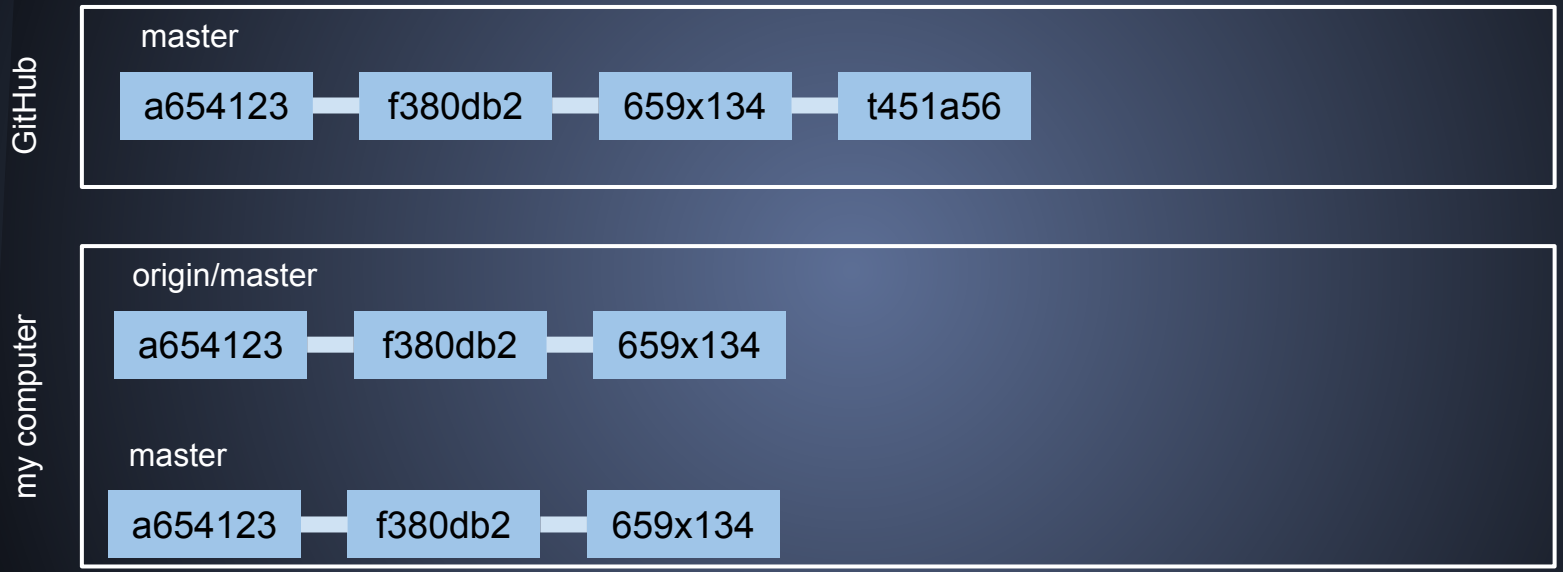
GitHub


GitHub
master
a654123
f380db2
659x134
git push
my computer
origin/master
a654123
f380db2
master
a654123
f380db2
659x134

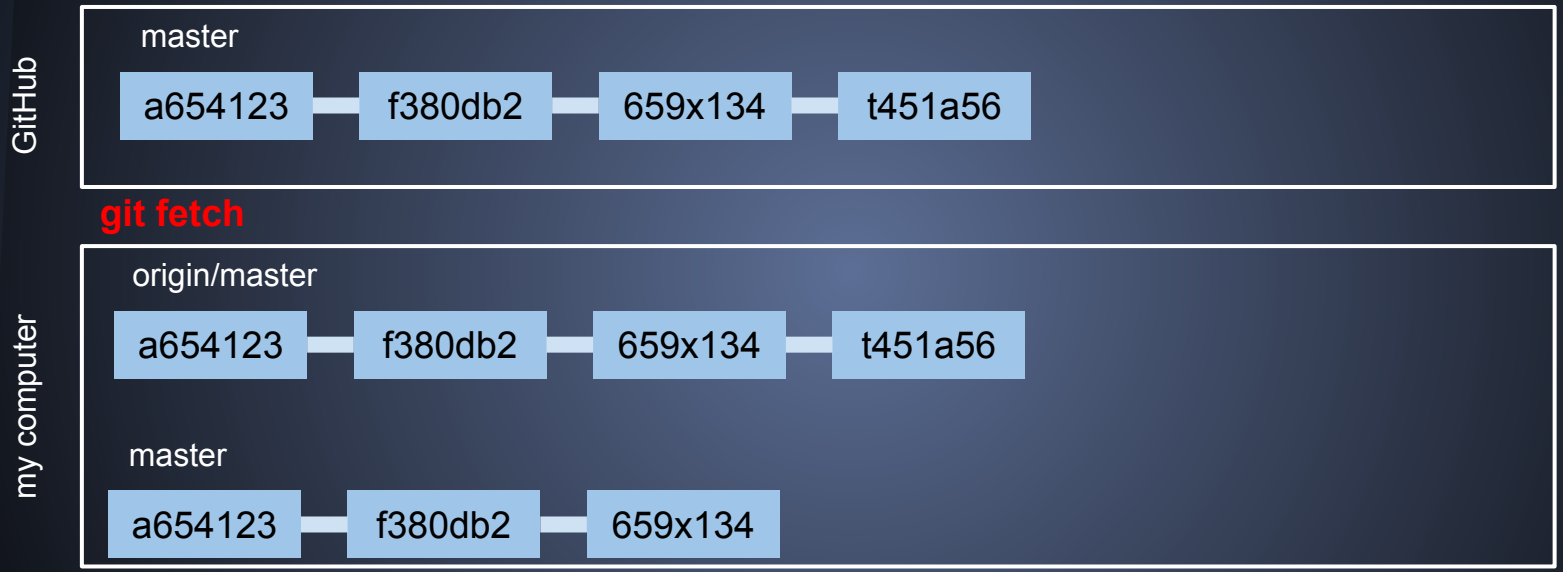
GitHub


GitHub
master
a654123
f380db2
659x134
my computer
origin/master - references remote repository and tries to stay in sync with it
a654123
f380db2
659x134
master
a654123
f380db2
659x134

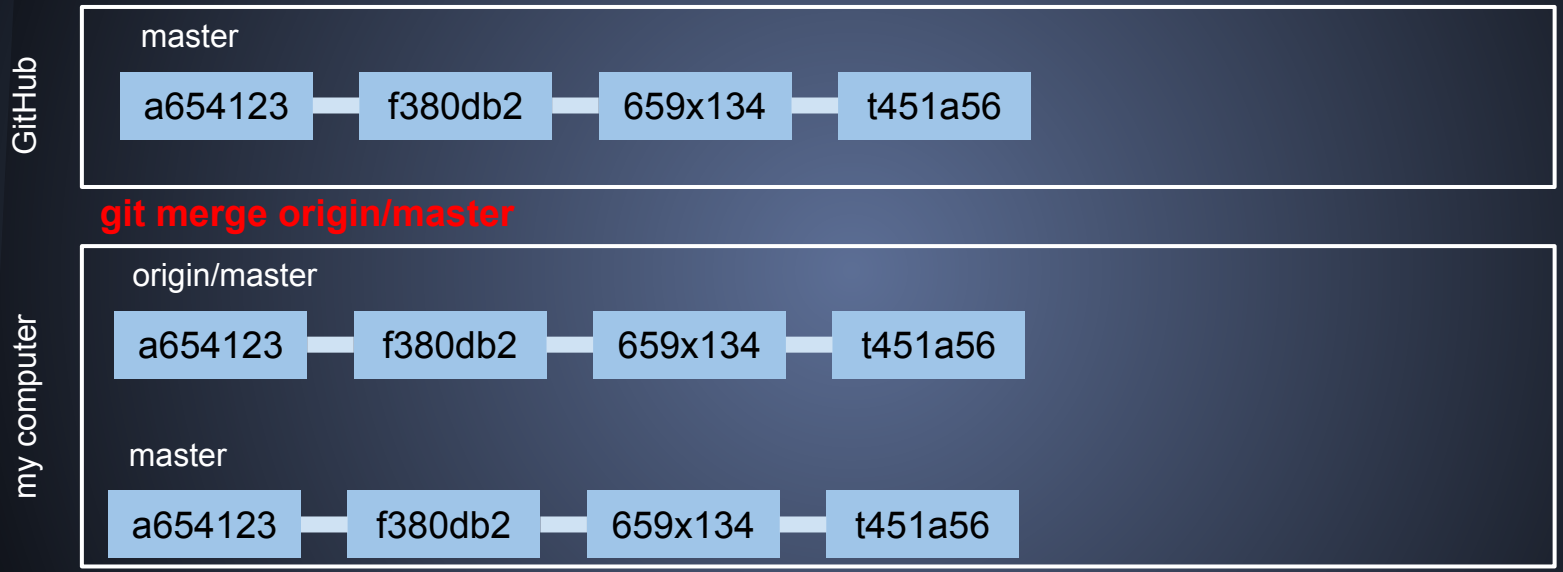
GitHub - somebody else contributes



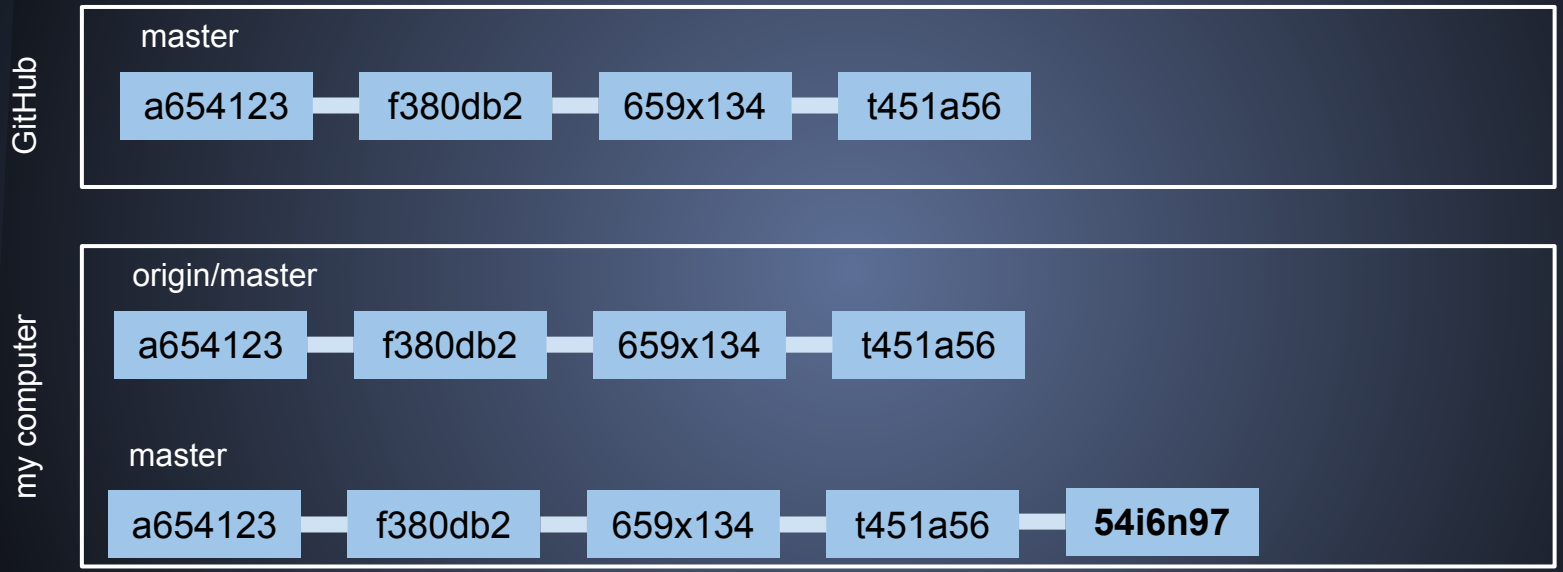
GitHub - somebody else contributes



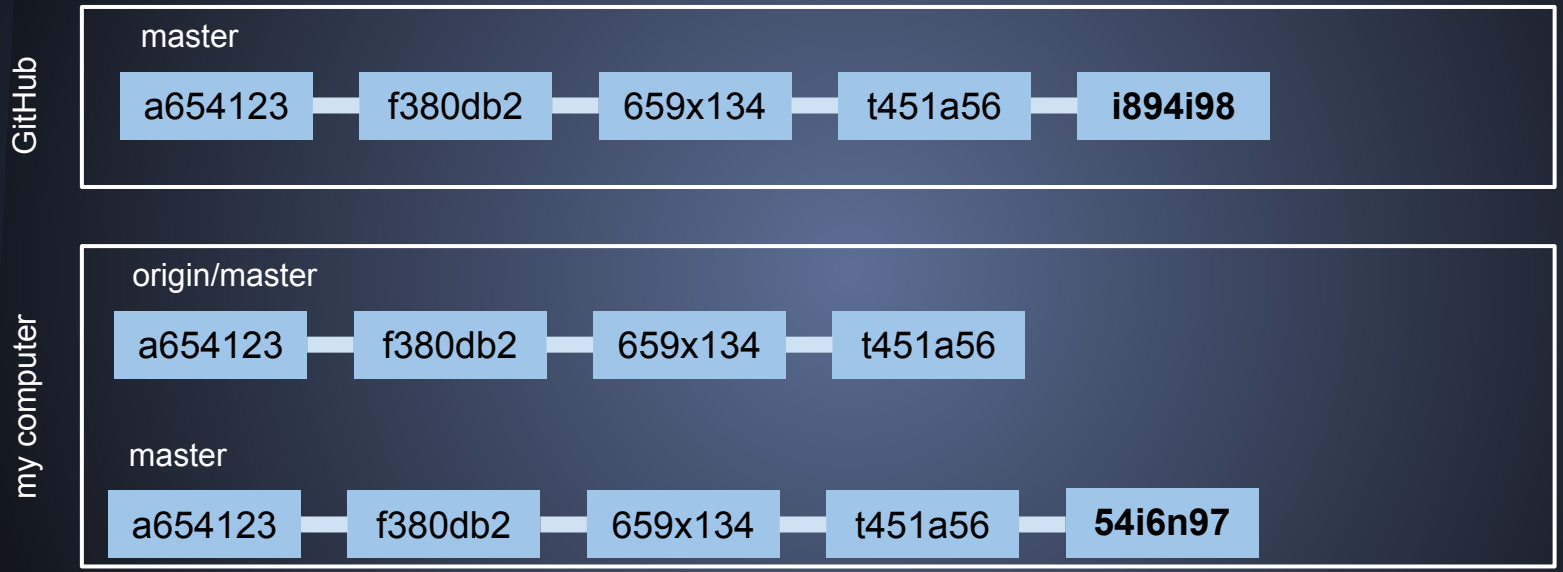
GitHub - somebody else contributes



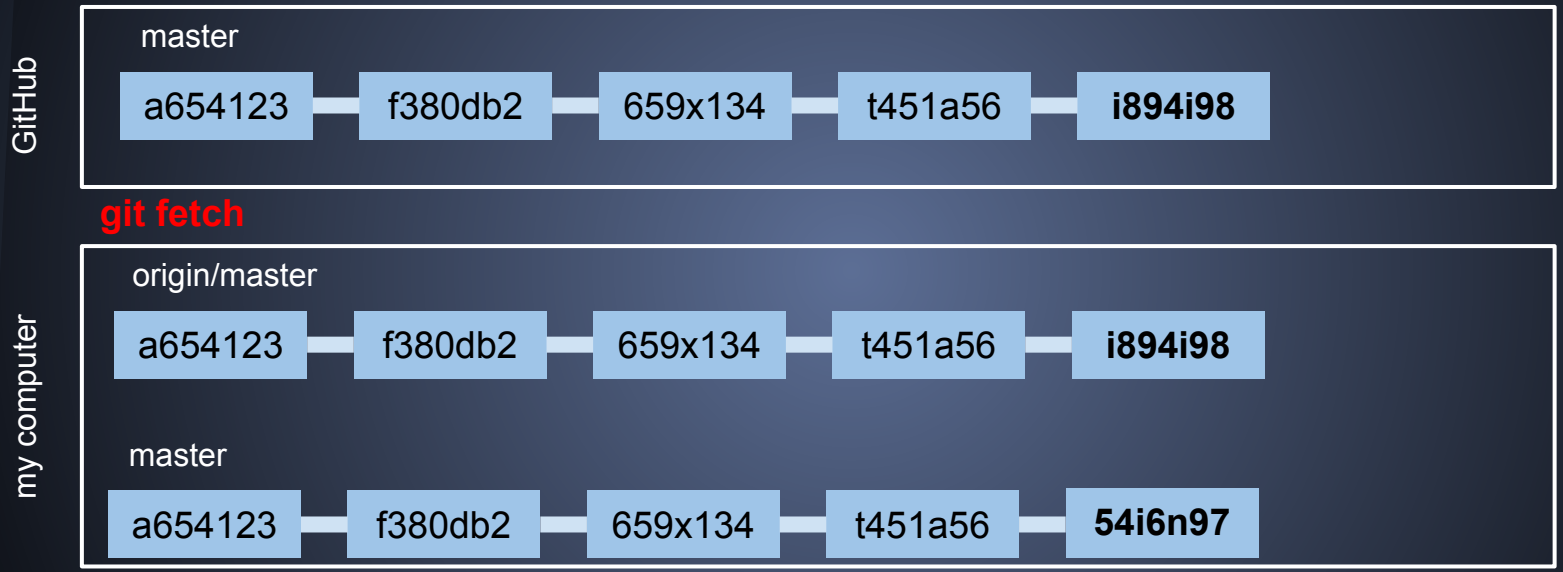
GitHub - ***I make changes***



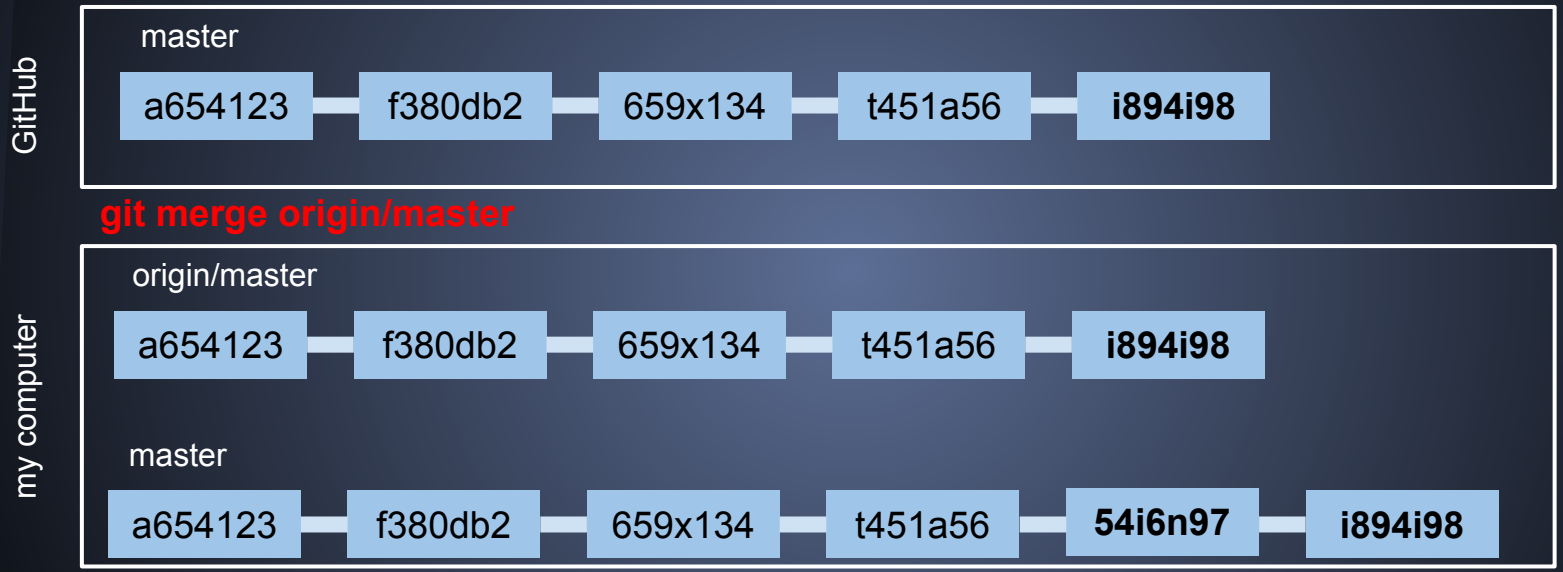
GitHub - somebody else contributes



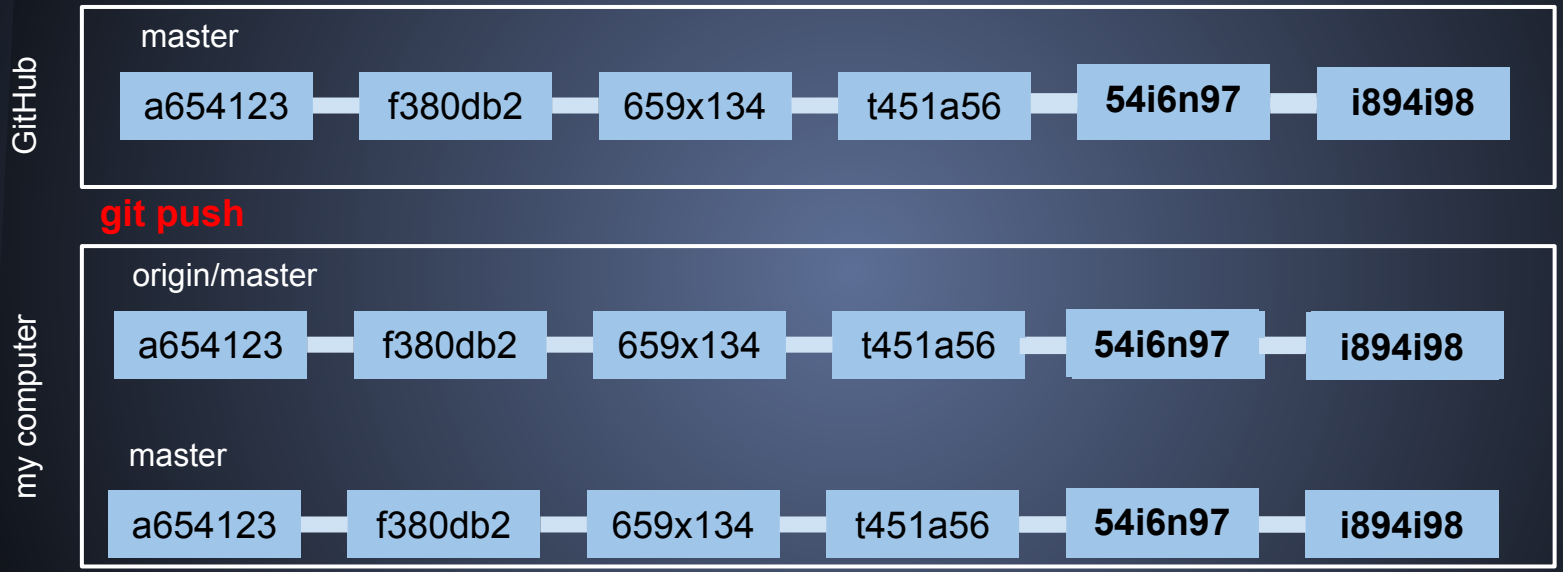
GitHub - somebody else contributes



GitHub - somebody else contributes



GitHub - somebody else contributes



GitHub - Workflow

- you're working locally: working directory, staging index, repository
  - changing files, adding files, committing files
    - (git add <filename>), (git commit -m "<message>")
  - this is occuring on your master branch

- you git fetch the latest from the remote repository
- you git merge your origin/master with your master branch
- you git push your master branch back to the remote repository

message best practices

- short
- descriptive
- what it does, not what you did

message best practices

- short
  - single-line summary (< 50 characters)
    - optional: follow with blank line & more complete description
- descriptive
  - no: “made changes”
  - yes: “login page allows google+ login”
- what it does, not what you did
  - no: “I made the login page to allow google+ login”
  - yes: “login page allows google+ login”

message best practices

- short
  - single-line summary (< 50 characters)
    - optional: follow with blank line & more complete description
- descriptive
  - no: “made changes”
  - yes: “login page allows google+ login”
- what it does, not what you did
  - no: “I made the login page to allow google+ login”
  - yes: “login page allows google+ login”
- present tense
  - no: “fixed bug”
  - yes: “fixes recursion infinite-loop error for Norwegian users”

common commands

git --version

git config

git config --list

git config <item from --list>

git help

git help <command>

git init

common commands

git add <filename>
git add .
git commit -m “<msg>”
git commit -am “<msg>”

git status

git log
git log --oneline
git log --oneline -3
git log <branch> --oneline -3

git log --since=“2013-04-27”
git log --until=“2014-01-30”
git log --author=“Todd”
git log <SHA>..<SHA>
git log <SHA>..<file_name>
git log -p <SHA>..<SHA>
git log --stat --summary
git log --oneline --graph --all --decorate

common commands

git diff

git diff <file_name>

git diff --oneline

git diff --staged

git diff --color-words <file_name>

git diff <SHA>

git diff <SHA> <file_name>

git diff <SHA>..<SHA>

git diff <SHA>..<SHA> <file_name>

git diff <SHA>..HEAD

git diff <SHA>..HEAD^^

git diff --stat --summary <SHA>..HEAD

git diff -w <SHA>..HEAD

git diff <branch>..<branch>

git diff --color-words <branch>..<branch>

shows the differences between our working directory and the repository, unless you use options, for example, the --staged option shows the difference between the staging index and the repository; the <SHA> option shows the diff between the working directory and the repository commit referenced by the <SHA>

common commands

git mv <old_name> <new_name>

git rm <file_name>

common commands

git checkout -- <file_name>

git checkout <SHA> -- <file_name>

the '--' means we're not checking out a branch but a file; this checkout is used to replace a file in our working directory with a file from the repository

git reset HEAD <file_name>

moves a file from the staging index back to the working directory

common commands

git revert <SHA>

undoes the changes of a commit

git commit --amend -m “new msg”

allows you to edit the last commit in the repository

common commands

git clean -n

git clean -f

-n previews untracked files which will be deleted
-f deletes untracked files

.gitignore

```
tempfile.txt
*.zip
*.pdf
log/*.log
log/*.log.[0-9]
assets/videos/
assets/imgs/*.psd
```

create .gitignore in the root of your project
https://github.com/github/gitignore
https://help.github.com/articles/ignoring-files

common commands

git ls-tree <tree-ish>

git show <SHA>

git show HEAD

view commits

common commands

git branch

git branch <some_name> creates branch

git checkout <branch_name>

git branch --merged

git branch -r

git branch -a

git branch --move old_name new_name

common commands

git fetch

git merge

git push

git remote

git remote -v

git remote add <alias> <url>

git remote rm <alias>

dashes

two for a long flag, one for a short flag

git comes from the linux world and follows the ‘standard’ convention for flags in linux

resources

Stanford’s guide to git

Unix & Mac Autocompletion

that's an "oh" not a "zero"

```
cd ~
curl -OL https://github.com/git/git/raw/master/contrib/completion/git-completion.bash
mv ~/git-completion.bash ~/.git-completion.bash
one of these:
        EDIT ~/.bash_profile
        EDIT ~/.bashrc
then ...
        add this code:
                if [ -f ~/.git-completion.bash ]; then
                        source ~/.git-completion.bash
                fi
```

uses the curl tool to download the autocompletion script into our home directory

rename

add it to our login configuration .. use bash_profile unless you know what bashrc is, then use whichever you like … EDIT using nano … for example, at cmd: **nano .bash_profile**

if the file exists, add the file to settings

Showing branch in command prompt

MAC/UNIX

- install completion script
- prompt stored in PS1 (prompt string 1)
- echo $PS1
- export PS1='anything'
- export PS1='\W$(__git_ps1 "(%s)") > '

edit .bash_profile (or bashrc)

```
if [ -f ~/.git-completion.bash ]; then
        source ~/.git-completion.bash
        export PS1='\W$(__git_ps1 "(%s)") > '
fi
```

Windows

- prompt stored in PS1 (prompt string 1)
- echo $PS1
- edit .bash_profile
  - if the file doesn't exist, create it
  - save it in your user directory
- edit .bash_profile
  - export PS1='\W$(__git_ps1 "(%s)") > '
- command line: source ~/.bash_profile

Ars longa, vita brevis

Hippocrates

more training

remove deleted files

```
git diff --diff-filter=D --name-only -z | xargs -0 git rm
```